RECEIVED
CENTRAL FAX CENTER
JAN 23 2006

PATTERSON & SHERIDAN, LLP
ATTORNEYS AT LAW

3040 Post Oak Blvd, Suite 1500
Houston, TX 77056-6582
TEL 713.623.4844
FAX 713.623.4846

FACSIMILE COVER SHEET

DATE: JANUARY 23, 2006

FILE NO: ROC920010348US1 (IBMK10348)

TO: MAIL STOP APPEAL BRIEF – PATENTS
Examiner Mark P. Francis

FAX NO: 1-571-273-8300

FROM: Gero G. McClellan

PAGE(S) with cover: 22

RE:

TITLE: DEBUG OF CODE WITH SELECTIVE DISPLAY OF DATA
U.S. SERIAL NO.: 10/090,341
FILING DATE: March 4, 2002
INVENTOR(S): Bates et al.
EXAMINER: Mark P. Francis
GROUP ART UNIT: 2124
CONFIRMATION NO.: 3890

Attached are the following document(s) for the above-referenced application:

Appeal Brief.

PAGE 1/22 * RCVD AT 1/23/2006 6:24:36 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/33 * DNIS:2738300 * CSID:7136234846 * DURATION (mm-ss):05-54

RECEIVED
CENTRAL FAX CENTER 002/022

JAN 2 3 2006

PATENT
Atty. Dkt. No. ROC920010348US1
PS Ref. No.: IBMK10348

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:
Bates et al.

Serial No.: 10/090,341

Filed: March 4, 2002

For: DEBUG OF CODE WITH SELECTIVE DISPLAY OF DATA

§ Confirmation No.: 3890

§ Group Art Unit: 2124

§ Examiner: Mark P. Francis

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner Mark P. Francis, on the date shown below:

January 23, 2006
Date

Jon Stewart

Dear Sir:

76

APPEAL BRIEF

Applicants submit this Appeal Brief to the Board of Patent Appeals and Interferences on appeal from the decision of the Examiner of Group Art Unit 2165 dated July 7, 2005, finally rejecting claims 1-8, 11-14 and 16-31. The final rejection of Claims 1-8, 11-14 and 16-31 is appealed.

Applicants submit this Appeal Brief to the Board of Patent Appeals and Interferences on appeal from the decision of the Examiner of Group Art Unit 2124 dated July 27, 2005, finally rejecting claims 1, 2, 4-6, 8-12, and 14-20. Applicants appeal the final rejection of claims 1, 2, 4-6, 8-12, and 14-20. This Appeal Brief is believed to be timely since facsimile transmitted by the due date of January 23, 2006, as set by mailing a Notice of Appeal on November 21, 2005. Please charge the fee of $500.00 for filing this brief to Deposit Account No. 09-0465/ROC920010348US1.

01/25/2006 EFLORES 00000092 090465 10090341
01 FC:1402 500.00 DA

430434_1

TABLE OF CONTENTS

Real Party In Interest

The present application has been assigned to International Business Machines Corporation, Armonk, New York.

Related Appeals and Interferences

Applicant asserts that no other appeals or interferences are known to the Applicant, the Applicant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 1-2, 4-6, 8-12 and 14-20 are pending in the application. Claims 1-20 were originally presented in the application. Claims 3, 7 and 13 have been canceled without prejudice. Claims 1, 2, 4-6, 8-12 and 14-20 stand finally rejected as discussed below. The final rejection of claims 1-2, 4-6, 8-12 and 14-20 is appealed. The pending claims are shown in the attached Claims Appendix.

Status of Amendments

All claim amendments have been entered by the Examiner. No amendments to the claims were proposed after the final rejection.

Summary of Claimed Subject Matter

Claimed embodiments include provide a method (see e.g., claim 1), and computer readable medium (see e.g., 9 and 16) for debugging executable code configured to access associated data in a data repository. See e.g., *Application*, ¶ 1, 8, 23, Abstract. That is, the executable code being debugged includes routines that access data stored in a data repository. *Id.*

Claimed embodiments include a method (see e.g., claims 1, 2, 4-6, 8) and a computer readable medium (see e.g., claims 9-11, 13-4) for selectively displaying data during debugging. See *Application*, ¶ 8, 23, 41-45, Figure 8. The claimed embodiments include initiating a debugging session for the executable code. See e.g., *Application*, ¶ 8, 32, 42, Figure 8. Once initiated, the method includes monitoring the step-by-step execution of the executable code. See e.g., *Application*, ¶ 8, 33-35, 40, 44, Figure 8, 804-818. The method also includes determining whether the monitored executable code has accessed the associated data in the data repository. See e.g., *Application* ¶ 8, 45, Figure 8, 810. If so, the method includes, determining whether to display the associated data on the basis of whether the associated data is restricted data. See e.g., *Application*, ¶ 8, 36-37, 45 Figure 4, 144, Figure 8, 818. As claimed, determining whether to display the associated data comprises referencing predefined access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed to unauthorized users. *Id.* If it is determined to not display the associated data on the basis of the referenced predefined access restriction rules, the method includes outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code. See e.g., *Application*, ¶ 29, 41, 36, 38, 39, 45, Figure 8, 820.

Claimed embodiments also include a computer readable medium (see e.g., claims 17-20) containing a debug program which, when executed, performs an operation of debugging code. The code is configured to access associated data in a

repository. The claimed debug program is configured to provide debugger user interface. See *Application*, ¶ 8, 23, 32-35 41-45, Figure 4, Figure 8. The debug program is also configured with a debug engine configured to selectively pass data to the debugger user interface according to predefined access restriction rules defining at least one rule prohibiting at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code. See e.g., *Application*, ¶ 8, 29, 36-39, 41-45 Figure 8.

Grounds of Rejection to be Reviewed on Appeal

1. Claims 1-2, 4-6, 8-12 and 14-20 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Wimble et al.*, U.S. Pat. No. 5,812,850 (hereinafter *Wimble*) in view of *Kim et al.*, U.S. Pat. No. 6,026,362 (hereinafter *Kim*).

ARGUMENTS

Obviousness of claims 1-2, 4-6, 8-12 and 14-20 over *Wimble* in view of *Klm*.

The Examiner bears the initial burden of establishing a *prima facie* case of obviousness. *See* MPEP § 2142. To establish a *prima facie* case of obviousness three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one ordinary skill in the art to modify the reference or to combine the reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP § 2143. The present rejection fails to establish at least the first and third criteria.

The References

Wimble discloses "a debugging system which provides an interactive and dynamic environment for computer program debugging." *Wimble*, 1:18-20. "The debugging system uses a database of information relating machine executable code to source code. The database is developed during the compilation process using an extensible object-oriented set of tools." *Wimble*, Abstract. The debugger disclosed in *Wimble* "is a tool which provides such a new class of debugging services." *Wimble*, 6:28-29. The debugger disclosed in *Wimble* "generates *information to be used by the debugger* in a Debugging System. The information may be in the form of data bases. Debugging information is really a database of information about a compiled program." *Wimble*, 6:30-34. *Wimble* is very specific regarding the debugger database. Specifically, Wimble discloses that information in the debugger database includes "a collection of Symbolic Elements 417 which is to be built, and maintain *a description of the program once it has been built*. *Wimble*, 8:27-37. As the emphasized passage makes clear, the debugger database of *Wimble* is created, and used, by the debugger to assist in the debugging process, and is not accessed by the program being debugged.

Kim discloses a method for debugging complex computer applications that includes "displaying the relationship between processes and resources of the processes and the contents of the stack and registers for threads of processes. Double clicking on the displayed information causes more detailed information to be displayed. The display is updated when predetermined operations are performed during debug." *Kim*, Abstract.

Kim also discloses that users may be required to "log on to the host on which the processes to be debugged are present." *Kim*, 14:41-43. For example, *Kim*, Figure 32, depicts an exemplary log on screen with a text box for a user to enter a user name and password. In operation, the "user enters one or more host names (for example, the host names of computers A and B of FIG. 12) and supplies his user name and password. The debug client sends this information to the host system, which validates the user name and password. If the user name and password are valid, the host computer initiates a debug server on the host system." *Kim*, 14:44-50. Importantly, once a user logs on to the debugging system of *Kim*, *Kim* does not disclose any further actions that restrict the activity of a user, or the data accessed by the program being debugged during a debugging system.

Kim also discloses a highly flexible and configurable user interface. As disclosed in *Kim*, users may expand or contract the amount of information that is displayed on screen at any one time. Specifically, *Kim* provides:

> In the course of a debugging session, a programmer will want to concentrate his or her efforts in a particular area of the program. The present invention allows the programmer to hide or show necessary program detail by allowing intermediate nodes of the call tree to be "collapsed" (as illustrated in FIG. 6) or "expanded" (as illustrated in FIG. 8).

Kim 11:48-53. However, any information that the programmer chooses to hide may be retrieved and subsequently displayed at the programmer's discretion. The passage above continues:

> To allow for such a control, each node which has child nodes is provided with an icon button which functions as a switch. When a node is collapsed, the child

> function nodes will not be displayed; when expanded, the child function nodes are displayed. *By allowing the programmer to control the level of detail displayed*, the debugger is providing the capability to control the visual complexity that the user has to deal with.

Kim 11:54-67.

The Examiner has steadfastly argued that *Wimble*, in view of *Kim*, discloses the claimed embodiments of the Applicants' invention directed to debugging executable code that is configured to access associated data in a data repository. That is, when the code being debugged access data from a data repository as part of the execution of the executable code. Further, the Examiner asserts that the flexible interface disclosed in *Kim* allowing the user to expand or contract the amount of information that is displayed on screen at any one time discloses the claimed limitations directed to preventing restricted data from being displayed, for example, by "outputting masking characters ... indicative of the associated data without revealing a value of the associated data". See *Claims* 1, 16. Applicants respectfully submit, however, that this argument mischaracterizes what is disclosed by both *Wimble* and *Kim*.

Regarding claims 1 and 9, the Examiner asserts that *Wimble* discloses "a method of debugging executable code configured to access associated data in a data repository." See *Final Office Action*, p. 2-3. However, the Examiner cites two passages from *Wimble* that are directed to a database used by a debugger to store debugging information regarding a program being debugged. As described above, using the debugger disclosed in *Wimble*, the executable code (i.e., the program being debugged or "compiled program") does not access associated data from the debugger database – the debugger does. Respectfully, the Examiner's argument is confusing, as there is no readily apparent reason why the program being debugged would ever access the debugging information used by the debugger. The whole purpose of debugging is to carefully monitor the execution of the program being debugged to help identify and correct programming bugs, without interfering with the operations of the program. Thus Applicants submit that while *Wimble* discloses a debugger configured to create a database of debugging information, this fails to disclose the recited limitation of

debugging executable code that is itself *configured to access associated data in a data repository*.

Further, it directly follows that *Wimble* does not disclose the recited limitation of determining whether the monitored executable code has accessed the associated data in the data repository. Quite the contrary, it would make so sense for the system of *Wimble* to do so. The debugger of *Wimble* would not create the debugger database and then monitor whether the very same debugger accesses the debugger database. Access by the debugger to the debugger database would be presumed. Otherwise, the debugger would not bother to create it. On this basis alone, Applicants submit that the rejection should be vacated and the claims be allowed.

Furthermore, *Wimble* fails to disclose the limitation of determining whether the monitored executable code has accessed the associated data in the data repository, and if so, determining whether to display the associated data on the basis of whether the associated data is restricted data, as recited by claims 1 and 9. Claimed embodiments recite limitations to a debugger that may be used debug executable code configured to access data from a database in accordance with a set of "access restriction rules" that prevent a user interacting with the debugger from accessing "restricted data" in a data repository. The Examiner argues that "*Kim* discloses a debug server that performs a security check to ensure that the user is authorized to initiate a requested process such as the name of a program file to be debugged" See *Advisory Action*, Continuation Sheet; *Final Office Action*, p. 3-4.

Respectfully, the well known actions of having users enter a username and password at a login screen simply fails to have any relationship to the limitations recited by claims 1 and 9. Specifically, claims 1 and 9 recite "determining whether to display the associated data comprises referencing predefined access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed to unauthorized users." The step is directed to monitoring the actions of a program being debugged during a debugging session in its actions of accessing data

from a data repository. These actions are simply unrelated to actions of a user logging in through a graphical user interface.

Finally, the Examiner asserts that because *Kim* discloses an interface that allows a programmer to selectively display, or to not display, certain information about a program being debugged. See *Final Office Action*, p. 3-4. However, as described above, the interface of Kim allows material to be shown or hiding based on nothing more than a particular user's whims.

In no way does this passage describe the recited limitation of prohibiting the display of the associated data (e.g., by using "masking characters," as recited by claim 1); rather, a user interacting with the debugging system of *Kim* remains free to retrieve and display and data regarding the program being debugged at any time. Quite simply, the features of *Kim* that allow a programmer to keep a desktop "clean and pretty" do not disclose the limitation of "*preventing* at least a portion of the associated data from being displayed to unauthorized users," as recited by claims 1 and 9. Under the system of *Kim*, the user is never, in any way, *prevented* from accessing data the user has chosen to temporarily move from the display. Rather, any time the user wishes to display information not currently shown, the user may interact with the interface to display such data. Thus, Applicants submit that *Kim* fails to the limitations of claims 1 and 9, as suggested by the Examiner.

Accordingly, for all the reasons set forth above, *Wimble*, in view of *Kim*, fails to teach or suggest the limitations recited by claims 1 and 9. Therefore, applicants submit that claims 1 and 9, along with dependent claims 2, 4-6, 8, 10-12 and 14-15, are allowable, and Applicants respectfully request that these claims be allowed.

Regarding claim 16, the Examiner asserts that *Wimble*, in view of *Kim*, discloses a computer-readable medium containing a debug program which, when executed, performs an operation of debugging code configured to access associated data in a repository. However, Applicants assert that *Wimble*, in view of *Kim*, fails to disclose at least the recited limitation of "a debug engine configured to selectively pass data to the debugger user interface according to predefined access restriction rules defining at

least one rule prohibiting at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code."

The Examiner concedes that *Wimble* fails to disclose this limitation, but asserts that *Kim* does. For all the reasons given above, however, Applicants submit that the passages cited from *Kim* fail to teach or suggest a debugger interface configured to prohibit "at least a portion of the associated data from being displayed," as recited by claim 16; rather, these passages from *Kim* disclose a debugger user interface that allows a user to alter what data is displayed at any particular time.

Accordingly, Applicants assert that *Wimble*, in view of *Kim*, fails to teach or suggest the limitations recited by claim 16. Therefore, Applicants submit that claim 16, along with claims 17-20, are allowable, and Applicants respectfully request that these claims be allowed.

CONCLUSION

The Examiner errs in finding that claims 1-2, 4-6, 8-12 and 14-20 are unpatentable over *Wimble* in view of *Kim* under 35 U.S.C. § 103(a). Withdrawal of the rejection and allowance of all claims is respectfully requested.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No., 44,227/
Gero G. McClellan
Registration No. 44,227
Patterson & Sheridan, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Appellants

CLAIMS APPENDIX

1. (Previously Presented) A method of debugging executable code configured to access associated data in a data repository, comprising:

initiating a debugging session for the executable code, and, during the debugging session:

monitoring step-by-step execution of the executable code;

determining whether the monitored executable code has accessed the associated data in the data repository;

if so, determining whether to display the associated data on the basis of whether the associated data is restricted data; wherein determining whether to display the associated data comprises referencing predefined access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed to unauthorized users; and

upon determining not to display the associated data on the basis of the referenced predefined access restriction rules, outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code.

2. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data comprises determining whether the associated data can be provided to a debugger user interface.

3. (Canceled) The method of claim 1, further comprising:

determining that the associated data cannot be displayed during the debugging session; and

outputting text characters on an output screen indicative of the associated data without revealing a value of the associated data.

4. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data comprises referencing a restricted data table created in

response to reading the associated data from the repository and according to the predefined access restriction rules.

5. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data is performed by a debugging program.

6. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data is performed by a debugging program implementing the predefined access restriction rules.

7. (Canceled) The method of claim 1, wherein determining whether the associated data can be displayed comprises referencing access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed.

8. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data comprises referencing a parse expression defining a data format and an output expression defining a restricted portion of the parse expression.

9. (Previously Presented) A computer-readable medium containing a debug program which, when executed, performs an operation comprising, during debugging of executable code:

monitoring step-by-step execution of the executable code, wherein the executable code is configured to access associated data in a repository;

determining whether the executable code has accessed the associated data in the repository; and

determining whether the associated data can be displayed on the basis of whether the associated data is restricted data; wherein determining whether the associated data can be displayed comprises referencing predefined access restriction rules defining at least one rule preventing at least a portion of the associated data from

being displayed to unauthorized users, whereby selected data from the data repository is concealed from a user debugging the executable code.

10. (Original) The computer-readable medium of claim 9, wherein determining whether the associated data can be displayed comprises determining whether the associated data can be provided to a debugger user interface.

11. (Original) The computer-readable medium of claim 9, further comprising:
determining that the associated data cannot be displayed; and
outputting text characters on an output screen indicative of the associated data without revealing a value of the associated data.

12. (Previously Presented) The computer-readable medium of claim 9, wherein determining whether the associated data can be displayed comprises referencing a restricted data table created in response to reading the associated data from the repository and according to the predefined access restriction rules.

13. (Canceled) The computer-readable medium of claim 9, wherein determining whether the associated data can be displayed comprises referencing access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed.

14. (Original) The computer-readable medium of claim 9, wherein determining whether the associated data can be displayed comprises referencing a parse expression defining a data format and an output expression defining a restricted portion of the parse expression.

15. (Previously Presented) The computer-readable medium of claim 9, wherein the executable code accesses the associated data comprising a record and wherein determining whether the associated data can be displayed comprises:

referencing the predefined access restriction rules defining at least one rule preventing at least one field value from being displayed; and

determining whether the record contains the at least one field value.

16. (Previously Presented) A computer-readable medium containing a debug program which, when executed, performs an operation of debugging code configured to access associated data in a repository, the debug program comprising:

a debugger user interface; and

a debug engine configured to selectively pass data to the debugger user interface according to predefined access restriction rules defining at least one rule prohibiting at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code.

17. (Previously Presented) The computer-readable medium of claim 16, wherein the debug engine is configured to:

determine that the associated data cannot be displayed during the debugging session; and

conceal the display of the associated data by displaying text characters on an output screen indicative of the associated data without revealing a value of the associated data.

18. (Original) The computer-readable medium of claim 16, wherein the debug engine is configured to selectively pass data to the debugger user interface by referencing a restricted data table created in response to reading the associated data from the repository and according to the predefined access restriction rules.

19. (Original) The computer-readable medium of claim 16, wherein the at least one rule defines a value and an associated value, wherein if the associated value has been displayed the debug engine will not provide the value to the debugger user interface for display.

20. (Original) The computer-readable medium of claim 16, wherein the at least one rule defines a parse expression defining a data format and an output expression defining a restricted portion of the parse expression, whereby all values having restricted portion will not be provided to the debugger user interface for display.